

# FIGURE 1

5 AGTCCCAGACGGGCTTTTCCCAGAGAGCTAAAAGAGAAGGGCCAGAGAATGTCGTCCCAG  
 CCAGCAGGGGAACCAGACCTCCCCCGGGGCCACAGAGGACTACTCCTATGGCAGCTGGTAC  
 ATCGATGAGCCCCAGGGGGGCGAGGAGCTCCAGCCAGAGGGGAAGTGCCCTCCTGCCAC  
 ACCAGCATAACCACCCGGCCTGTACCACGCTGCTGGCCTCGCTGTCAATCCTTGTGCTG  
 CTGCTCCTGGCCATGTGGTGAGGCGCCGCCAGCTCTGGCCTGACTGTGTGCGTGGCAGG  
 CCGGGCCTGCCAGCCCTGTGGATTCTTGGCTGGGGACAGGCCCCGGGCAGTGCCCTGCT  
 10 GCTGTTTTCATGGTCTCCTGAGCTCCCTGTGTTTGTGCTCCCCGACGAGGACGCATTG  
 CCTTCTGACTCTCGCCTCAGCACCCAGCCAAGATGGGAAAAGTGAAGCTCCAAGAGGG  
 GCCTGGAAGATACTGGGACTGTTCTATTATGTGCCCTCTACTACCCTCTGGCTGCCTGT  
 GCCACGGCTGGCCACACAGCTGCACACCTGCTCGGCAGCACGCTGTCTGGGCCACCTT  
 GGGGTCCAGGTCTGGCAGAGGGCAGAGTGTCCCCAGGTGCCCAAGATCTACAAGTACTAC  
 15 TCCCTGCTGGCCTCCCTGCCTCTCCTGTGGGCTCGGATTCTTGAGCCTTTGGTACCCT  
 GTGCAGCTGGTGAGAAGCTTCAGCCGTAGGACAGGAGCAGGCTCCAAGGGGCTGCAGAGC  
 AGCTACTCTGAGGAATATCTGAGGAACCTCCCTTTCAGGAAGAAGCTGGGAAGCAGCTAC  
 CACACCTCCAAGCATGGCTTCCTGTCTGGGCCCGCGTCTGCTTGAGACACTGCATCTAC  
 ACTCCACAGCCAGGATTCCATCTCCCGCTGAAGCTGGTGTCTTCAGCTACACTGACAGGG  
 20 ACGGCCATTTACCAGGTGGCCCTGTGCTGTGGTGGGCGTGGTACCCACTATCCAGAAG  
 GTGAGGGCAGGGGTACCCACGGATGTCTCCTACCTGCTGGCCGGCTTTGGAATCGTGCTC  
 TCCGAGGACAAGCAGGAGGTGGTGGAGCTGGTGAAGCACCATCTGTGGGCTCTGGAAGTG  
 TGCTACATCTCAGCCTTGGTCTTGTCTGCTTACTCACCTTCTGGTCTCTGATGCGCTCA  
 CTGGTGACACACAGGACCAACCTTCGAGCTCTGCACCGAGGAGCTGCCCTGGACTTGAGT  
 25 CCCTTGCACTCGGAGTCCCCATCCCTCCCGCCAAGCCATATTCTGTTGGATGAGCTTCAGT  
 GCCTACCAGACAGCCTTTATCTGCCTTGGGCTCCTGGTGACGAGATCATCTTCTTCTG  
 GGAACCACGGCCCTGGCCTTCCTGGTGTCTATGCCTGTGCTCCATGGCAGGAACCTCCTG  
 CTCTTCCGTTCCCTGGAGTCTCTGGCCCTTCTGGCTGACTTTGGCCCTGGCTGTGATC  
 CTGCAGAACATGGCAGCCCATTTGGGTCTTCTGGAGACTCATGATGGACACCCACAGCTG  
 30 ACCAACCGGCGAGTGCTCTATGCAGCCACCTTTCTTCTTCTTCCCCCTCAATGTGCTGGTG  
 GGTGCCATGGTGGCCACCTGGCGAGTGCTCCTCTCTGCCCTCTACAACGCCATCCACCTT  
 GGCCAGATGGACCTCAGCCTGCTGCCACCGAGAGCCGCCACTCTCGACCCCGGCTACTAC  
 ACGTACCGAACTTCTTGAAGATTGAAGTCAGCCAGTCGCATCCAGCCATGACAGCCTTC  
 TGCTCCCTGCTCCTGCAAGCGCAGAGCCTCTACCCAGGACCATGGCAGCCCCCAGGAC  
 35 AGCCTCAGACCAGGGGAGGAAGACGAAGGGATGCAGCTGCTACAGACAAAGGACTCCATG  
 GCCAAGGGAGCTAGGCCCCGGGGCCAGCCGCGGCAGGGCTCGCTGGGGTCTGGCCTACAG  
 CTGCTGCACAACCCAACCTGCAGGTCTTCGCAAGACGGCCCTGTTGGGTGCCAATGGT  
 GCCCAGCCCTGAGGGCAGGGAAGGTCAACCCACCTGCCCATCTGTGCTGAGGCATGTTCC  
 TGCCTACCATCCTCCTCCTCCCCGGCTCTCCTCCCAGCATCACACCAGCCATGCAGCCA  
 40 GCAGGTCTCCGGATCACTGTGGTTGGGTGGAGGTCTGTCTGCACTGGGAGCCTCAGGAG  
 GGCTCTGCTCCACCCACTTGGCTATGGGAGAGCCAGCAGGGGTCTGGAGAAAAAACTG  
 GTGGGTAGGGCCTTGGTCCAGGAGCCAGTTGAGCCAGGGCAGCCACATCCAGGCGTCTC  
 CCTACCCTGGCTCTGCCATCAGCCTTGAAGGCCCTCGATGAAGCCTTCTCTGGAACCACT  
 CCAGCCCAGCTCCACCTCAGCCTTGGCCTTCACGCTGTGGAAGCAGCCAAGGCACTTCTT  
 45 CACCCCCCAGCGCCACGGACCTCTCTGGGGAGTGCCCGAAAGCTCCCGGTCTCTGGC  
 CTGCAGGGCAGCCCAAGTCATGACTCAGACCAGGTCCACACTGAGCTGCCACACTCGA  
 GAGCCAGATATTTTTGTAGTTTTTATGCCTTTGGCTATTATGAAAGAGGTTAGTGTGTTT  
 CCTGCAATAAACTGTTCCTGAGAAAAA  
 50

## FIGURE 2

MSSQPAGNQTS PGATEDYSYGSWYIDEPQGGEELQPEGEVPSCHTSIPPGLYHACLASLS  
ILVLLLLLAMLVRRRQLWPDVCVRGRPLSPVDFLAGDRPRAVPAAVFMVLLSSLCLLLDP  
EDALPFLTLASAPSQDGKTEAPRGAWKILGLFYAALYYPLAACATAGHTAAHLLGSTLS  
5 WAHLGVQVWQRAECPQVPKIYKYSSLLASLPLLLGLGFLSLWYFVQLVRSFSRRTGAGSK  
GLQSSYSEYLRNLLCRKKLGSSYHTSKHGFLSWARVCLRHCIYTPQPGFHLPLKLVLSA  
TLTGTAIYQVALLLVGVVPTIQKVRAGVTTDVSYLLAGFGIVLSEDKQEVVELVKHHLW  
ALEVCYISALVLSCLLTFLVLMRSLVTHRTNLRALHRGAALDLSPLHRSPHPSRQAIFCW  
MSFSAYQTAFICLGLLVQQIIFFLGTTALAFVLMPVLHGRNLLFRSLESSWPFWLTLA  
10 LAVILQNMAAHWVFLETHDGHPQLTNRRVLYAATFLLFPLNVLVGAMVATWRVLLSALYN  
AIHLGQMDLSLLPPRAATLDPGYTYRNFLKIEVSQSHPAFTAFCSTLLQAQSLLPRTMA  
APQDSLRLPGEEDGEMQLLQTKDSMAKGARPGASGRARWGLAYTLLHNPTLQVFRKTALL  
GANGAQP

Important features of the protein:

15 Signal peptide:

None

Transmembrane domain:

20

54-69

102-119

148-166

207-222

25

301-320

364-380

431-451

474-489

560-535

30

Motif file:

Motif name: N-glycosylation site.

8-12

35

Motif name: N-myristoylation site.

50-56

176-182

40

241-247

317-323

341-347

525-531

627-633

45

631-637

640-646

661-667

Motif name: Prokaryotic membrane lipoprotein lipid attachment site.

50

364-375

Motif name: ATP/GTP-binding site motif A (P-loop).

55

132-140

## FIGURE 3A

PRO	XXXXXXXXXXXXXXXXXX	(Length = 15 amino acids)
Comparison Protein	XXXXXXYYYYYYY	(Length = 12 amino acids)

5

% amino acid sequence identity =

(the number of identically matching amino acid residues between the two polypeptide sequences as determined by ALIGN-2) divided by (the total number of amino acid residues of the PRO polypeptide) =

10

5 divided by 15 = 33.3%

**FIGURE 3B**

PRO	XXXXXXXXXX	(Length = 10 amino acids)
Comparison Protein	XXXXXXYYYYYYZZYZ	(Length = 15 amino acids)

5

% amino acid sequence identity =

(the number of identically matching amino acid residues between the two polypeptide sequences as determined by ALIGN-2) divided by (the total number of amino acid residues of the PRO polypeptide) =

10

5 divided by 10 = 50%

	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415	2416	2417	2418	2419	2420	2421	2422	2
--	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	---

## FIGURE 3C

PRO-DNA                      NNNNNNNNNNNNNNNN                      (Length        =        14  
nucleotides)

5    Comparison DNA                      NNNNNNLLLLLLLLLL                      (Length        =        16  
nucleotides)

% nucleic acid sequence identity =

10    (the number of identically matching nucleotides between the two nucleic acid sequences as  
determined by ALIGN-2) divided by (the total number of nucleotides of the PRO-DNA  
nucleic acid sequence) =

6 divided by 14 = 42.9%

15

## FIGURE 3D

PRO-DNA	NNNNNNNNNNNNNN	(Length = 12 nucleotides)
Comparison DNA	NNNNLLLTVV	(Length = 9 nucleotides)

5

% nucleic acid sequence identity =

(the number of identically matching nucleotides between the two nucleic acid sequences as determined by ALIGN-2) divided by (the total number of nucleotides of the PRO-DNA nucleic acid sequence) =

10

4 divided by 12 = 33.3%

## FIGURE 4A

```

/*
 *
 * C-C increased from 12 to 15
5  * Z is average of EQ
 * B is average of ND
 * match with stop is _M; stop-stop = 0; J (joker) match = 0
 */
10 #define _M      -8      /* value of a match with a stop */

int      _day[26][26] = {
/*      A B C D E F G H I J K L M N O P Q R S T U V W X Y Z */
/* A */      { 2, 0, -2, 0, 0, -4, 1, -1, -1, 0, -1, -2, -1, 0, _M, 1, 0, -2, 1, 1, 0, 0, -6, 0, -3, 0},
/* B */      { 0, 3, -4, 3, 2, -5, 0, 1, -2, 0, 0, -3, -2, 2, _M, -1, 1, 0, 0, 0, 0, -2, -5, 0, -3, 1},
15 /* C */      {-2, -4, 15, -5, -5, -4, -3, -3, -2, 0, -5, -6, -5, -4, _M, -3, -5, -4, 0, -2, 0, -2, -8, 0, 0, -5},
/* D */      { 0, 3, -5, 4, 3, -6, 1, 1, -2, 0, 0, -4, -3, 2, _M, -1, 2, -1, 0, 0, 0, -2, -7, 0, -4, 2},
/* E */      { 0, 2, -5, 3, 4, -5, 0, 1, -2, 0, 0, -3, -2, 1, _M, -1, 2, -1, 0, 0, 0, -2, -7, 0, -4, 3},
/* F */      {-4, -5, -4, -6, -5, 9, -5, -2, 1, 0, -5, 2, 0, -4, _M, -5, -5, -4, -3, -3, 0, -1, 0, 0, 7, -5},
/* G */      { 1, 0, -3, 1, 0, -5, 5, -2, -3, 0, -2, -4, -3, 0, _M, -1, -1, -3, 1, 0, 0, -1, -7, 0, -5, 0},
20 /* H */      {-1, 1, -3, 1, 1, -2, -2, 6, -2, 0, 0, -2, -2, 2, _M, 0, 3, 2, -1, -1, 0, -2, -3, 0, 0, 2},
/* I */      {-1, -2, -2, -2, -2, 1, -3, -2, 5, 0, -2, 2, 2, -2, _M, -2, -2, -2, -1, 0, 0, 4, -5, 0, -1, -2},
/* J */      { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, _M, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
/* K */      {-1, 0, -5, 0, 0, -5, -2, 0, -2, 0, 5, -3, 0, 1, _M, -1, 1, 3, 0, 0, 0, -2, -3, 0, -4, 0},
/* L */      {-2, -3, -6, -4, -3, 2, -4, -2, 2, 0, -3, 6, 4, -3, _M, -3, -2, -3, -3, -1, 0, 2, -2, 0, -1, -2},
25 /* M */      {-1, -2, -5, -3, -2, 0, -3, -2, 2, 0, 0, 4, 6, -2, _M, -2, -1, 0, -2, -1, 0, 2, -4, 0, -2, -1},
/* N */      { 0, 2, -4, 2, 1, -4, 0, 2, -2, 0, 1, -3, -2, 2, _M, -1, 1, 0, 1, 0, 0, -2, -4, 0, -2, 1},
/* O */      { _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M},
0, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M, _M},
/* P */      { 1, -1, -3, -1, -1, -5, -1, 0, -2, 0, -1, -3, -2, -1, _M, 6, 0, 0, 1, 0, 0, -1, -6, 0, -5, 0},
30 /* Q */      { 0, 1, -5, 2, 2, -5, -1, 3, -2, 0, 1, -2, -1, 1, _M, 0, 4, 1, -1, -1, 0, -2, -5, 0, -4, 3},
/* R */      {-2, 0, -4, -1, -1, -4, -3, 2, -2, 0, 3, -3, 0, 0, _M, 0, 1, 6, 0, -1, 0, -2, 2, 0, -4, 0},
/* S */      { 1, 0, 0, 0, 0, -3, 1, -1, -1, 0, 0, -3, -2, 1, _M, 1, -1, 0, 2, 1, 0, -1, -2, 0, -3, 0},
/* T */      { 1, 0, -2, 0, 0, -3, 0, -1, 0, 0, 0, -1, -1, 0, _M, 0, -1, -1, 1, 3, 0, 0, -5, 0, -3, 0},
/* U */      { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, _M, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
35 /* V */      { 0, -2, -2, -2, -2, -1, -1, -2, 4, 0, -2, 2, 2, -2, _M, -1, -2, -2, -1, 0, 0, 4, -6, 0, -2, -2},
/* W */      {-6, -5, -8, -7, -7, 0, -7, -3, -5, 0, -3, -2, -4, -4, _M, -6, -5, 2, -2, -5, 0, -6, 17, 0, 0, -6},
/* X */      { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, _M, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
/* Y */      {-3, -3, 0, -4, -4, 7, -5, 0, -1, 0, -4, -1, -2, -2, _M, -5, -4, -4, -3, -3, 0, -2, 0, 0, 10, -4},
40 /* Z */      { 0, 1, -5, 2, 3, -5, 0, 2, -2, 0, 0, -2, -1, 1, _M, 0, 3, 0, 0, 0, 0, -2, -6, 0, -4, 4}
};

```

45

50

55

## FIGURE 4B

```

/*
*/
#include <stdio.h>
5  #include <ctype.h>

#define MAXJMP      16      /* max jumps in a diag */
#define MAXGAP      24      /* don't continue to penalize gaps larger than this */
#define JMPS        1024    /* max jmps in an path */
10  #define MX        4      /* save if there's at least MX-1 bases since last jmp */

#define DMAT         3      /* value of matching bases */
#define DMIS         0      /* penalty for mismatched bases */
#define DINS0        8      /* penalty for a gap */
15  #define DINS1        1    /* penalty per base */
#define PINS0        8      /* penalty for a gap */
#define PINS1        4      /* penalty per residue */

struct jmp {
20      short          n[MAXJMP]; /* size of jmp (neg for dely) */
      unsigned short  x[MAXJMP]; /* base no. of jmp in seq x */
}; /* limits seq to 2^16-1 */

struct diag {
25      int            score;      /* score at last jmp */
      long            offset;     /* offset of prev block */
      short           jmp;        /* current jmp index */
      struct jmp      jp;         /* list of jmps */
};

30  struct path {
      int             spc;         /* number of leading spaces */
      short           n[JMPS]; /* size of jmp (gap) */
      int             x[JMPS]; /* loc of jmp (last elem before gap) */
35  };

char      *ofile;      /* output file name */
char      *name[2];    /* seq names. getseqs() */
char      *prog;        /* prog name for err msgs */
40  char      *seq[2];   /* seqs: getseqs() */
int        dmax;        /* best diag: nw() */
int        dmax0;       /* final diag */
int        dna;         /* set if dna: main() */
int        endgaps;     /* set if penalizing end gaps */
45  int        gapx, gapy; /* total gaps in seqs */
int        len0, len1;  /* seq lens */
int        ngapx, ngapy; /* total size of gaps */
int        smax;        /* max score: nw() */
int        *xbm;        /* bitmap for matching */
50  long       offset;    /* current offset in jmp file */
struct     diag      *dx; /* holds diagonals */
struct     path      pp[2]; /* holds path for seqs */

char      *calloc(), *malloc(), *index(), *strcpy();
55  char      *getseq(), *g_calloc();

```



## FIGURE 4C

```

/* Needleman-Wunsch alignment program
*
* usage: prog file1 file2
5  * where file1 and file2 are two dna or two protein sequences.
* The sequences can be in upper- or lower-case and may contain ambiguity
* Any lines beginning with ';', '>' or '<' are ignored
* Max file length is 65535 (limited by unsigned short x in the jmp struct)
* A sequence with 1/3 or more of its elements ACGTU is assumed to be DNA
10 * Output is in the file "align.out"
*
* The program may create a tmp file in /tmp to hold info about traceback.
* Original version developed under BSD 4.3 on a vax 8650
*/
15 #include "nw.h"
#include "day.h"

static _dbval[26] = {
20 1,14,2,13,0,0,4,11,0,0,12,0,3,15,0,0,0,5,6,8,8,7,9,0,10,0
};

static _pbval[26] = {
25 1, 2|(1<<('D'-'A'))|(1<<('N'-'A')), 4, 8, 16, 32, 64,
128, 256, 0xFFFFFFFF, 1<<10, 1<<11, 1<<12, 1<<13, 1<<14,
1<<15, 1<<16, 1<<17, 1<<18, 1<<19, 1<<20, 1<<21, 1<<22,
1<<23, 1<<24, 1<<25|(1<<('E'-'A'))|(1<<('Q'-'A'))
};

main(ac, av)                                main
30     int    ac;
     char    *av[];
{
    prog = av[0];
    if (ac != 3) {
35         fprintf(stderr, "usage: %s file1 file2\n", prog);
         fprintf(stderr, "where file1 and file2 are two dna or two protein sequences.\n");
         fprintf(stderr, "The sequences can be in upper- or lower-case\n");
         fprintf(stderr, "Any lines beginning with ';' or '<' are ignored\n");
         fprintf(stderr, "Output is in the file \"align.out\"\n");
40         exit(1);
    }
    namex[0] = av[1];
    namex[1] = av[2];
    seqx[0] = getseq(namex[0], &len0);
45    seqx[1] = getseq(namex[1], &len1);
    xbm = (dna)? _dbval : _pbval;

    endgaps = 0;                                /* 1 to penalize endgaps */
    ofile = "align.out";                        /* output file */

50    nw();                                /* fill in the matrix, get the possible jmps */
    readjmps();                                /* get the actual jmps */
    print();                                /* print stats, alignment */

55    cleanup(0);                                /* unlink any tmp files */
}

```

## FIGURE 4D

```

/* do the alignment, return best score: main()
 * dna: values in Fitch and Smith, PNAS, 80, 1382-1386, 1983
 * pro: PAM 250 values
5  * When scores are equal, we prefer mismatches to any gap, prefer
 * a new gap to extending an ongoing gap, and prefer a gap in seqx
 * to a gap in seq y.
 */
nw()
10 {
    char      *px, *py;      /* seqs and ptrs */
    int        *ndely, *dely; /* keep track of dely */
    int        ndelx, delx;   /* keep track of delx */
    int        *tmp;         /* for swapping row0, row1 */
15  int        mis;          /* score for each type */
    int        ins0, ins1;    /* insertion penalties */
    register   id;           /* diagonal index */
    register   ij;           /* jmp index */
    register   *col0, *col1;  /* score for curr, last row */
20  register   xx, yy;        /* index into seqs */

    dx = (struct diag *)g_calloc("to get diags", len0 + len1 + 1, sizeof(struct diag));

    ndely = (int *)g_calloc("to get ndely", len1 + 1, sizeof(int));
25  dely = (int *)g_calloc("to get dely", len1 + 1, sizeof(int));
    col0 = (int *)g_calloc("to get col0", len1 + 1, sizeof(int));
    col1 = (int *)g_calloc("to get col1", len1 + 1, sizeof(int));
    ins0 = (dna)? DINS0 : PINS0;
    ins1 = (dna)? DINS1 : PINS1;

    smax = -10000;
    if (endgaps) {
        for (col0[0] = dely[0] = -ins0, yy = 1; yy <= len1; yy++) {
30          col0[yy] = dely[yy] = col0[yy-1] - ins1;
          ndely[yy] = yy;
        }
        col0[0] = 0; /* Waterman Bull Math Biol 84 */
    }
    else
40      for (yy = 1; yy <= len1; yy++)
        dely[yy] = -ins0;

    /* fill in match matrix
    */
45  for (px = seqx[0], xx = 1; xx <= len0; px++, xx++) {
    /* initialize first entry in col
    */
    if (endgaps) {
        if (xx == 1)
50          col1[0] = delx = -(ins0 + ins1);
        else
          col1[0] = delx = col0[0] - ins1;
        ndelx = xx;
    }
    else {
55      col1[0] = 0;
        delx = -ins0;
        ndelx = 0;
    }
60  }

```

# FIGURE 4E

...nw

```

5      for (py = seqx[1], yy = 1; yy <= len1; py ++, yy ++ ) {
        mis = col0[yy-1];
        if (dna)
            mis += (xbm[*px-'A']&xbm[*py-'A'])? DMAT : DMIS;
        else
            mis += _day[*px-'A'][*py-'A'];

10      /* update penalty for del in x seq;
        * favor new del over ongoing del
        * ignore MAXGAP if weighting endgaps
        */
        if (endgaps || ndely[yy] < MAXGAP) {
15            if (col0[yy] - ins0 >= dely[yy]) {
                dely[yy] = col0[yy] - (ins0 + ins1);
                ndely[yy] = 1;
            } else {
                dely[yy] -= ins1;
                ndely[yy] ++;
            }
        } else {
            if (col0[yy] - (ins0 + ins1) >= dely[yy]) {
25                dely[yy] = col0[yy] - (ins0 + ins1);
                ndely[yy] = 1;
            } else
                ndely[yy] ++;
        }

30      /* update penalty for del in y seq;
        * favor new del over ongoing del
        */
        if (endgaps || ndelx < MAXGAP) {
            if (col1[yy-1] - ins0 >= delx) {
35                delx = col1[yy-1] - (ins0 + ins1);
                ndelx = 1;
            } else {
                delx -= ins1;
                ndelx ++;
            }
        } else {
            if (col1[yy-1] - (ins0 + ins1) >= delx) {
40                delx = col1[yy-1] - (ins0 + ins1);
                ndelx = 1;
            } else
45                ndelx ++;
        }

        /* pick the maximum score; we're favoring
        * mis over any del and delx over dely
        */
50
55
60

```

# FIGURE 4F

...DW

```

id = xx - yy + len1 - 1;
if (mis >= delx && mis >= dely[yy])
    col1[yy] = mis;
5   else if (delx >= dely[yy]) {
        col1[yy] = delx;
        ij = dx[id] ijmp;
        if (dx[id].jp.n[0] && (!dna || (ndelx >= MAXJMP
10   && xx > dx[id].jp.x[ij] + MX) || mis > dx[id].score + DINS0)) {
            dx[id].ijmp++;
            if (++ij >= MAXJMP) {
                writejumps(id);
                ij = dx[id] ijmp = 0;
                dx[id].offset = offset;
                offset += sizeof(struct jmp) + sizeof(offset);
            }
        }
        dx[id] jp.n[ij] = ndelx;
        dx[id] jp.x[ij] = xx;
        dx[id] score = delx;
    }
    else {
        col1[yy] = dely[yy];
        ij = dx[id].ijmp;
25   if (dx[id] jp.n[0] && (!dna || (ndely[yy] >= MAXJMP
        && xx > dx[id] jp.x[ij] + MX) || mis > dx[id] score + DINS0)) {
            dx[id].ijmp++;
            if (++ij >= MAXJMP) {
                writejumps(id);
                ij = dx[id].ijmp = 0;
                dx[id].offset = offset;
                offset += sizeof(struct jmp) + sizeof(offset);
            }
        }
        dx[id] jp.n[ij] = -ndely[yy];
        dx[id] jp.x[ij] = xx;
        dx[id] score = dely[yy];
    }
40   if (xx == len0 && yy < len1) {
        /* last col
        */
        if (endgaps)
            col1[yy] -= ins0 + ins1*(len1-yy);
        if (col1[yy] > smax) {
            smax = col1[yy];
            dmax = id;
        }
    }
50   }
    if (endgaps && xx < len0)
        col1[yy-1] -= ins0 + ins1*(len0-xx);
    if (col1[yy-1] > smax) {
        smax = col1[yy-1];
        dmax = id;
    }
    tmp = col0; col0 = col1; col1 = tmp;
60   }
    (void) free((char *)ndely);
    (void) free((char *)dely);
    (void) free((char *)col0); (void) free((char *)col1);}

```

## FIGURE 4G

```

/*
 *
 * print() -- only routine visible outside this module
 *
5  * static:
 * getmat() -- trace back best path, count matches: print()
 * pr_align() -- print alignment of described in array p[]: print()
 * dumpblock() -- dump a block of lines with numbers, stars: pr_align()
10 * nums() -- put out a number line: dumpblock()
 * putline() -- put out a line (name, [num], seq, [num]): dumpblock()
 * stars() -- put a line of stars: dumpblock()
 * stripname() -- strip any path and prefix from a seqname
 */
15
#include "nw.h"

#define SPC      3
#define P_LINE  256    /* maximum output line */
20 #define P_SPC   3      /* space between name or num and seq */

extern  _day[26][26];
int     olen;          /* set output line length */
FILE    *fx;           /* output file */
25

print()
{
    int     lx, ly, firstgap, lastgap;    /* overlap */

    if ((fx = fopen(ofile, "w")) == 0) {
        fprintf(stderr, "%s: can't write %s\n", prog, ofile);
        cleanup(1);
    }
    fprintf(fx, "< first sequence: %s (length = %d)\n", namex[0], len0);
    fprintf(fx, "< second sequence: %s (length = %d)\n", namex[1], len1);
    olen = 60;
    lx = len0;
    ly = len1;
    firstgap = lastgap = 0;
    if (dmax < len1 - 1) { /* leading gap in x */
        pp[0].spc = firstgap = len1 - dmax - 1;
        ly -= pp[0].spc;
    }
    else if (dmax > len1 - 1) { /* leading gap in y */
        pp[1].spc = firstgap = dmax - (len1 - 1);
        lx -= pp[1].spc;
    }
    if (dmax0 < len0 - 1) { /* trailing gap in x */
        lastgap = len0 - dmax0 - 1;
        lx -= lastgap;
    }
    else if (dmax0 > len0 - 1) { /* trailing gap in y */
        lastgap = dmax0 - (len0 - 1);
        ly -= lastgap;
    }
    getmat(lx, ly, firstgap, lastgap);
    pr_align();
}
60

```

print

## FIGURE 4H

```

/*
 * trace back the best path, count matches
 */
5 static
getmat(lx, ly, firstgap, lastgap)                                getmat
    int    lx, ly;                /* "core" (minus endgaps) */
    int    firstgap, lastgap;     /* leading trailing overlap */
{
10     int    nm, i0, i1, siz0, siz1;
    char    outx[32];
    double   pct;
    register n0, n1;
    register char *p0, *p1;

15     /* get total matches, score
    */
    i0 = i1 = siz0 = siz1 = 0;
    p0 = seqx[0] + pp[1].spc;
    p1 = seqx[1] + pp[0].spc;
    n0 = pp[1].spc + 1;
    n1 = pp[0].spc + 1;

20     nm = 0;
    while ( *p0 && *p1 ) {
        if (siz0) {
            p1++;
            n1++;
            siz0--;
30         }
        else if (siz1) {
            p0++;
            n0++;
            siz1--;
35         }
        else {
            if (xbm[*p0-'A'] & xbm[*p1-'A'])
                nm++;
            if (n0++ == pp[0].x[i0])
                siz0 = pp[0].n[i0++];
            if (n1++ == pp[1].x[i1])
                siz1 = pp[1].n[i1++];
            p0++;
            p1++;
40         }
    }

45     /* pct homology:
    * if penalizing endgaps, base is the shorter seq
    * else, knock off overhangs and take shorter core
    */
    if (endgaps)
        lx = (len0 < len1)? len0 : len1;
    else
50         lx = (lx < ly)? lx : ly;
    pct = 100.*(double)nm/(double)lx;
    fprintf(fx, "\n");
    fprintf(fx, "< %d match%s in an overlap of %d: %.2f percent similarity\n",
60         nm, (nm == 1)? "" : "es", lx, pct);

```

## FIGURE 4I

```

fprintf(fx, "< gaps in first sequence: %d", gapx);
if (gapx) {
    (void) sprintf(outx, " (%d %s%s)",
        ngapx, (dna)? "base":"residue", (ngapx == 1)? ":" : "s");
    fprintf(fx, "%s", outx);

    fprintf(fx, ", gaps in second sequence: %d", gapy);
    if (gapy) {
        (void) sprintf(outx, " (%d %s%s)",
            ngapy, (dna)? "base":"residue", (ngapy == 1)? ":" : "s");
        fprintf(fx, "%s", outx);
    }
    if (dna)
        fprintf(fx,
            "\n< score: %d (match = %d, mismatch = %d, gap penalty = %d + %d per base)\n",
            smax, DMAT, DMIS, DINS0, DINS1);
    else
        fprintf(fx,
            "\n< score: %d (Dayhoff PAM 250 matrix, gap penalty = %d + %d per residue)\n",
            smax, PINS0, PINS1);
    if (endgaps)
        fprintf(fx,
            "< endgaps penalized. left endgap: %d %s%s, right endgap: %d %s%s\n",
            firstgap, (dna)? "base" : "residue", (firstgap == 1)? "" : "s",
            lastgap, (dna)? "base" : "residue", (lastgap == 1)? "" : "s");
    else
        fprintf(fx, "< endgaps not penalized\n");
}

static      nn;          /* matches in core -- for checking */
static      lmax;        /* lengths of stripped file names */
static      ij[2];       /* jmp index for a path */
static      nc[2];       /* number at start of current line */
static      n[2];        /* current elem number -- for gapping */
static      siz[2];
static char *ps[2];      /* ptr to current element */
static char *po[2];      /* ptr to next output char slot */
static char out[2][P_LINE]; /* output line */
static char star[P_LINE]; /* set by stars() */

/*
 * print alignment of described in struct path pp[]
 */
static
pr_align()
{
    int      nn;          /* char count */
    int      more;
    register i;

    for (i = 0, lmax = 0; i < 2; i++) {
        nn = stripname(name[i]);
        if (nn > lmax)
            lmax = nn;

        nc[i] = 1;
        ni[i] = 1;
        siz[i] = ij[i] = 0;
        ps[i] = seqx[i];
        po[i] = out[i];
    }
}

```

...getmat

pr\_align

## FIGURE 4J

```

for (nn = nm = 0, more = 1; more; ) {
    for (i = more = 0; i < 2; i++) {
        /*
         * do we have more of this sequence?
         */
        if (!*ps[i])
            continue;

        more++;

        if (pp[i].spc) { /* leading space */
            *po[i]++ = ' ';
            pp[i].spc--;
        }
        else if (siz[i]) { /* in a gap */
            *po[i]++ = '-';
            siz[i]--;
        }
        else { /* we're putting a seq element
            */
            *po[i] = *ps[i];
            if (islower(*ps[i]))
                *ps[i] = toupper(*ps[i]);
            po[i]++;
            ps[i]++;

            /*
             * are we at next gap for this seq?
             */
            if (ni[i] == pp[i].x[ij[i]]) {
                /*
                 * we need to merge all gaps
                 * at this location
                 */
                siz[i] = pp[i].n[ij[i] + +];
                while (ni[i] == pp[i].x[ij[i]])
                    siz[i] += pp[i].n[ij[i] + +];
            }
            ni[i]++;
        }
    }
    if (++nn == olen || !more && nn) {
        dumpblock();
        for (i = 0; i < 2; i++)
            po[i] = out[i];
        nn = 0;
    }
}

/*
 * dump a block of lines, including numbers, stars: pr_align()
 */
static
dumpblock()
{
    register i;

    for (i = 0; i < 2; i++)
        *po[i]-- = '\0';
}

```

...pr\_align

dumpblock



# FIGURE 4K

...dumpblock

```

5      (void) putc('\n', fx);
      for (i = 0; i < 2; i++) {
          if (*out[i] && (*out[i] != ' ' || *(po[i]) != ' ')) {
              if (i == 0)
                  nums(i);
              if (i == 0 && *out[1])
                  stars();
              putline(i);
              if (i == 0 && *out[1])
                  fprintf(fx, star);
              if (i == 1)
                  nums(i);
          }
      }
}

20  /*
    * put out a number line: dumpblock()
    */
    static
    nums(ix)
25      int      ix;      /* index in out[] holding seq line */
    {
        char      nline[P_LINE];
        register  i, j;
        register char *pn, *px, *py;
30
        for (pn = nline, i = 0; i < lmax + P_SPC; i++, pn++)
            *pn = ' ';
        for (i = nc[ix], py = out[ix]; *py; py++, pn++) {
            if (*py == ' ' || *py == '-')
                *pn = ' ';
35            else {
                if (i%10 == 0 || (i == 1 && nc[ix] != 1)) {
                    j = (i < 0)? -i : i;
                    for (px = pn; j; j /= 10, px--)
                        *px = j%10 + '0';
40                    if (i < 0)
                        *px = '-';
                }
                else
45                    *pn = ' ';
                i++;
            }
        }
        *pn = '\0';
        nc[ix] = i;
        for (pn = nline; *pn; pn++)
            (void) putc(*pn, fx);
        (void) putc('\n', fx);
50    }

55  /*
    * put out a line (name, [num], seq, [num]): dumpblock()
    */
    static
    putline(ix)
60      int      ix;
    {

```

nums

putline

## FIGURE 4L

...putline

```

5      int          i;
      register char *px;

      for (px = namex[ix], i = 0; *px && *px != ':'; px++, i++)
          (void) putc(*px, fx);
10     for (; i < lmax+P_SPC; i++)
          (void) putc(' ', fx);

      /* these count from 1:
       * ni[] is current element (from 1)
       * nc[] is number at start of current line
       */
15     for (px = out[ix]; *px; px++)
          (void) putc(*px&0x7F, fx);
      (void) putc('\n', fx);
20 }

/*
 * put a line of stars (seqs always in out[0], out[1]): dumpblock()
 */
25 static
stars()
{
    int          i;
    register char *p0, *p1, cx, *px;

30     if (!*out[0] || (*out[0] == ' ' && *(p0[0]) == ' ') ||
        !*out[1] || (*out[1] == ' ' && *(p0[1]) == ' '))
        return;
    px = star;
35     for (i = lmax+P_SPC; i; i--)
        *px++ = ' ';

    for (p0 = out[0], p1 = out[1]; *p0 && *p1; p0++, p1++) {
40         if (xbm[*p0-'A']&xbm[*p1-'A']) {
            cx = '*';
            nm++;
        }
45         else if (!dna && _day[*p0-'A'][*p1-'A'] > 0)
            cx = '.';
        else
            cx = ' ';

50         }
        else
            cx = ' ';
        *px++ = cx;
    }
    *px++ = '\n';
55     *px = '\0';
}

60

```

stars

## FIGURE 4M

```
/*
 * strip path or prefix from pn, return len: pr_align()
 */
5 static
  stripname(pn)                                stripname
  char      *pn;      /* file name (may be path) */
{
  register char      *px, *py;
10      py = 0;
      for (px = pn; *px; px++)
          if (*px == '/')
15              py = px + 1;
      if (py)
          (void) strcpy(pn, py);
      return(strlen(pn));
20  }

25

30

35

40

45

50

55

60
```

## FIGURE 4N

```

/*
 * cleanup() -- cleanup any tmp file
 * getseq() -- read in seq, set dna, len, maxlen
5  * g_calloc() -- calloc() with error checkin
 * readjumps() -- get the good jumps, from tmp file if necessary
 * writejumps() -- write a filled array of jumps to a tmp file: nw()
 */
10 #include "nw.h"
#include <sys/file.h>

char    *jname = "/tmp/homgXXXXXX";          /* tmp file for jumps */
FILE    *fj;

15 int    cleanup();                          /* cleanup tmp file */
long    lseek();

/*
20 * remove any tmp file if we blow
 */
cleanup(i)                                     cleanup
{
    int    i;

    if (fj)
25         (void) unlink(jname);
    exit(i);
}

/*
30 * read, return ptr to seq, set dna, len, maxlen
 * skip lines starting with ';', '<', or '>'
 * seq in upper or lower case
 */
35 char    *
getseq(file, len)                             getseq
{
    char    *file;          /* file name */
    int     *len;           /* seq len */

    {
        char    line[1024], *pseq;
        register char    *px, *py;
        int     natgc, tlen;
        FILE    *fp;

        if ((fp = fopen(file, "r")) == 0) {
45             fprintf(stderr, "%s: can't read %s\n", prog, file);
             exit(1);
        }
        tlen = natgc = 0;
        while (fgets(line, 1024, fp)) {
50             if (*line == ';' || *line == '<' || *line == '>')
                 continue;
             for (px = line; *px != '\n'; px++)
                 if (isupper(*px) || islower(*px))
                     tlen++;
55         }
        if ((pseq = malloc((unsigned)(tlen+6))) == 0) {
             fprintf(stderr, "%s: malloc() failed to get %d bytes for %s\n", prog, tlen+6, file);
             exit(1);
        }
        pseq[0] = pseq[1] = pseq[2] = pseq[3] = '\0';
60
    }
}

```

## FIGURE 40

...getseq

```

py = pseq + 4;
*len = tlen;
rewind(fp);

5
while (fgets(line, 1024, fp)) {
    if (*line == ';' || *line == '<' || *line == '>')
        continue;
10    for (px = line; *px != '\n'; px++) {
        if (isupper(*px))
            *py++ = *px;
        else if (islower(*px))
            *py++ = toupper(*px);
15    if (index("ATGCU",*(py-1)))
        natgc++;
    }
}
*py++ = '\0';
*py = '\0';
(void) fclose(fp);
dna = natgc > (tlen/3);
return(pseq+4);
25
char *
g_alloc(msg, nx, sz)
char *msg; /* program, calling routine */
int nx, sz; /* number and size of elements */
30 {
    char *px, *calloc();

    if ((px = calloc((unsigned)nx, (unsigned)sz)) == 0) {
        if (*msg) {
35            fprintf(stderr, "%s: g_alloc() failed %s (n=%d, sz=%d)\n", prog, msg, nx, sz);
            exit(1);
        }
    }
    return(px);
40 }

/*
 * get final jmps from dx[] or tmp file, set pp[], reset dmax: main()
 */
45 readjmps()
{
    int fd = -1;
    int siz, i0, i1;
    register i, j, xx;

50    if (fj) {
        (void) fclose(fj);
        if ((fd = open(jname, O_RDONLY, 0)) < 0) {
            fprintf(stderr, "%s: can't open() %s\n", prog, jname);
            cleanup(1);
55        }
    }
    for (i = i0 = i1 = 0, dmax0 = dmax, xx = len0; i++) {
        while (1) {
60            for (j = dx[dmax].ijmp; j >= 0 && dx[dmax].jp.x[j] >= xx; j--)
                ;

```

## FIGURE 4P

...readjumps

```

5      if (j < 0 && dx[dmax].offset && fj) {
        (void) lseek(fd, dx[dmax].offset, 0);
        (void) read(fd, (char *)&dx[dmax] jp, sizeof(struct jmp));
        (void) read(fd, (char *)&dx[dmax].offset, sizeof(dx[dmax].offset));
        dx[dmax] ijmp = MAXJMP-1;
      }
10     else
        break;
    }
    if (i >= JMPS) {
        fprintf(stderr, "%s: too many gaps in alignment\n", prog);
        cleanup(1);
    }
15     if (j >= 0) {
        siz = dx[dmax] jp.n[j];
        xx = dx[dmax] jp.x[j];
        dmax += siz;
        if (siz < 0) { /* gap in second seq */
            pp[1].n[i1] = -siz;
            xx += siz;

            /* id = xx - yy + len1 - 1
            */
            pp[1] x[i1] = xx - dmax + len1 - 1;
            gapy ++,
            ngapy -= siz;
        }
        /* ignore MAXGAP when doing endgaps */
        siz = (-siz < MAXGAP || endgaps)? -siz : MAXGAP;
        i1 ++,
    }
    else if (siz > 0) { /* gap in first seq */
        pp[0] n[i0] = siz;
        pp[0] x[i0] = xx;
        gapx ++,
        ngapx += siz;
        /* ignore MAXGAP when doing endgaps */
        siz = (siz < MAXGAP || endgaps)? siz : MAXGAP;
        i0 ++,
    }
    }
    else
        break;
45 }

/* reverse the order of jumps
*/
for (j = 0, i0--, j < i0; j ++, i0--) {
50     i = pp[0].n[j]; pp[0].n[j] = pp[0].n[i0]; pp[0].n[i0] = i;
    i = pp[0].x[j]; pp[0].x[j] = pp[0].x[i0]; pp[0].x[i0] = i;
}
for (j = 0, i1--; j < i1; j ++, i1--) {
55     i = pp[1].n[j]; pp[1].n[j] = pp[1].n[i1]; pp[1].n[i1] = i;
    i = pp[1].x[j]; pp[1].x[j] = pp[1].x[i1]; pp[1].x[i1] = i;
}
if (fd >= 0)
    (void) close(fd);
if (fj) {
60     (void) unlink(jname);
    fj = 0;
    offset = 0;}}

```

## FIGURE 4Q

```
/*
 * write a filled jmp struct offset of the prev one (if any): nw()
 */
writejumps(ix)
{
    int    ix;
    char    *mktemp0;

    if (!fj) {
        if (mktemp(jname) < 0) {
            fprintf(stderr, "%s: can't mktemp() %s\n", prog, jname);
            cleanup(1);
        }
        if ((fj = fopen(jname, "w")) == 0) {
            fprintf(stderr, "%s: can't write %s\n", prog, jname);
            exit(1);
        }
    }
    (void) fwrite((char *)&dx[ix].jp, sizeof(struct jmp), 1, fj);
    (void) fwrite((char *)&dx[ix].offset, sizeof(dx[ix].offset), 1, fj);
}
```

writejumps

## FIGURE 5

5 GTGCTCTCCGAGGACAAGCAGGAGGNGGTGGAGCTGGTGAAGCACCATCTGTGGGCTCTG  
GAAGTGTGCTACATCTCAGCCTTGGTCTTGCTTACTCACCTTCCTGGTCCCTGATG  
CGCTCACTGGTGACACACAGGACCAACCTTCGAGCTCTGCACCGAGGAGCTGCCCTGGAC  
10 TTGAGTCCCTTGCAATCGGAGTCCCCATCCCTCCCGCCAAGCCATATTCTGTTGGATGAGC  
TTCAGTGCCTACCCAGACAGCCTTTATCTGCCCTTGGGCTCCTGGTGCAGCAGATCATCTTC  
TTCCTGGGAACCCACGGCCCTGGCCTTCCTGGTGTCTCATGCCCTGTGCTCCATGGCAGGAAC  
CTCCTGCTCTTCCGTTCCCTGGAGTCCTCGTGGCCCTTCTGGCTGACTTTGGCCCTGGCT  
GTGATCCTGCAGAACATGGCAGCCCATTTGGGTCTTCCTGGAGACTCATGATGGACACCCA  
CAGCTGACCAACCGGCGAGTGTCTCTATGCAGCCACCTTCTTCTCTTCCCCCTCAATGTG  
15 CTGGTGGGTGCCATGGTGGCCACCTGGCGAGTGTCTCCTCTCTGCCCTCTACAACGCCATC  
CACCTTGGCCAGATGGACCTCAGCCTGTCTGCCACCGAGAGCCGCCACTCTCGACCCCGGC  
TACTACACGTACCGAA



## FIGURE 6

5 CACAACCAGCCACCCCTCTAGGATCCCAGCCCAGCTGGTGCTGGGCTCAGAGGAGAAGGC  
 CCCGTGTTGGGAGCACCCCTGCTTGCCCTGGAGGGACAAGTTTCCGGGAGAGATCAATAAAG  
 GAAAGGAAAGAGACAAGGAAGGGAGAGGTGAGGAGAGCGCTTGATTGGAGGAGAAGGGCC  
 AGAGAA**TG**TCGTCCCAGCCAGCAGGGAACCCAGACCTCCCCCGGGGCCACAGAGGACTACT  
 CCTATGGCAGCTGGTACATCGATGAGCCCCAGGGGGGCGAGGAGCTCCAGCCAGAGGGGG  
 10 AAGTGGCCCTCCTGCCACACCAGCATAACCACCCGGCCTGTACCACGCCTGCCTGGCCTCGC  
 TGTCAATCCTTGTGCTGCTGCTCCTGGCCATGCTGGTGAGGCGCCGCCAGCTCTGGCCTG  
 ACTGTGTGCGTGGCAGGCCCCGGCCTGCCAGGCCCCGGGCAGTGCCTGCTGCTGTTTTCA  
 TGGTCCTCCTGAGCTCCCTGTGTTTGCTGCTCCCCGACGAGGACGCATTGCCCTTCCTGA  
 CTCTCGCCTCAGCACCCAGCCAAGATGGGAAAAGTGAAGGCTCCAAGAGGGGCCCTGGAAGA  
 TACTGGGACTGTTCTATTATGCTGCCCTCTACTACCTCTGGCTGCCTGTGCCACGGCTG  
 15 GCCACACAGCTGCACACCTGCTCGGCAGCACGCTGTCTGGGCCACCTTGGGGTCCAGG  
 TCTGGCAGAGGGCAGAGTGTCCCCAGGTGCCCAAGATCTACAAGTACTACTCCCTGCTGG  
 CCTCCCTGCCTCTCCTGCTGGGCCTCGGATTCTTGAGCCTTTGGTACCCTGTGCAGCTGG  
 TGAGAAGCTTCAGCCGTAGGACAGGAGCAGGCTCCAAGGGGCTGCAGAGCAGCTACTCTG  
 AGGAATATCTGAGGAACCTCCTTTGCAGGAAGAAGCTGGGAAGCAGCTACCACACCTCCA  
 20 AGCATGGCTTCCTGTCCTGGGCCCCGCGTCTGCTTGAGACACTGCATCTACACTCCACAGC  
 CAGGATTCCATCTCCCGCTGAAGCTGGTGCTTTCAGCTACACTGACAGGGACGGCCATTT  
 ACCAGGTGGCCCTGCTGCTGCTGGTGGGCGTGGTACCCACTATCCAGAAGGTGAGGGCAG  
 GGGTCACCACGGATGTCTCCTACCTGCTGGCCGGCTTTGGAATCGTGCTCTCCGAGGACA  
 AGCAGGAGGTGGTGGAGCTGGTGAAGCACCATCTGTGGGCTCTGGAAGTGTGCTACATCT  
 25 CAGCCTTGGTCTTGTCTGCTTACTCACCTTCCTGGTCCTGATGCGCTCACTGGTGACAC  
 ACAGGACCAACCTTCGAGCTCTGCACCGAGGAGCTGCCCTGGACTTGAGTCCCTTGATC  
 GGAGTCCCCATCCCTCCCGCCAAGCCATATTCTGTTGGATGAGCTTCAGTGCCTACCAGA  
 CAGCCTTTATCTGCCTTGGGCTCCTGGTGCAGCAGATCATCTTCTTCTGGGAACACGG  
 CCCTGGCCTTCCTGGTGCTCATGCCTGTGCTCCATGGCAGGAACCTCCTGCTCTTCCGTT  
 30 CCCTGGAGTCCCTCGTGGCCCTTCTGGCTGACTTTGGCCCTGGCTGTGATCTGCAGAACA  
 TGGCAGCCCATTTGGGTCTTCTGGAGACTCATGATGGACACCCACAGCTGACCAACCGGC  
 GAGTGCTCTATGCAGCCACCTTTCTTCTCTTCCCCCTCAATGTGCTGGTGGGTGCCATAG  
 TGGCCACCTGGCGAGTGCTCCTCTCTGCCCTCTACAACGCCATCCACCTTGGCCAGATGG  
 ACCTCAGCCTGCTGCCACCGAGAGCCGCCACTCTCGACCCCGGCTACTACACGTACCGAA  
 35 ACTTCTTGAAAGATTGAAGTCAGCCAGTCGCATCCAGCCATGACAGCCTTCTGCTCCCTGC  
 TCCTGCAAGCGCAGAGCCTCCTACCCAGGACCATGGCAGCCCCCAGGACAGCCTCAGAC  
 CAGGGGAGGAAGACGAAGGGATGCAGCTGCTACAGACAAAGGACTCCATGGCCAAGGGAG  
 CTAGGCCCGGGGCCAGCCGCGGCAGGGCTCGCTGGGGTCTGGCCTACACGCTGCTGCACA  
 ACCCAACCCCTGCAGGTCTTCCGCAAGACGGCCCTGTTGGGTGCCAATGGTGCCCAGCCCT  
 40 GAGGGCAGGGAAGGTCAACCCACCTGCCATCTGTGCTGAGGCATGTTCTGCTACCAC  
 CTCCTCCCTCCCCGGCTCTCCTCCAGCATCACACCAGCCATGCAGCCAGCAGGTCCCTCC  
 GGATCACTGTGGTTGGGTGGAGGTCTGTCTGCACTGGGAGCCTCAGGAGGGCTCTGCTCC  
 ACCCACTTGGCTATGGGAGAGCCAGCAGGGGTCTGGAGAAAGAACTGGTGGGTTAGGG  
 CCTTGGTCCAGGAGCCAGTTGAGCCAGGGCAGCCACATCCAGGCGTCTCCCTACCCTGGC  
 45 TCTGCCATCAGCCTTGAAGGGCCTCGATGAAGCCTTCTCTGGAACCACTCCAGCCCAGCT  
 CCACCTCAGCCTTGGCCTTCACGCTGTGGAAGCAGCCAAGGCACTTCCTCACCCCTCAG  
 CGCCACGGACCTCTCTGGGGAGTGGCCGGAAAGCTCCCGGGCCTCTGGCCTGCAGGGCAG  
 CCCAAGTCATGACTCAGACCAGGTCCCACACTGAGCTGCCACACTCGAGAGCCAGATAT  
 TTTTGTAGTTTTTATGCCTTTGGCTATTATGAAAGAGGTTAGTGTGTTCCCTGCAATAAA  
 50 CTTGTTCCCTGAGAAAAA

## FIGURE 7

MSSQPAGNQTS PGATEDYSYGSWYIDEPQGGEELQPEGEVPSCHTSIPPGLYHACLASL  
SILVLLLLLAMLVRRRQLWPDCVRGRPGLP RPRAVPAAVFMVLLSSLCLLLPDEDALPFL  
5 TLASAPSQDGKTEAPRGAWKILGLFYAALYYPLAACATAGHTAAHLLGSTLSWAHLGV  
QVWQRAECPQVPKIYKYSSLLASLPLLLGLGFLSLWYPVQLVRSFSRRTGAGSKGLQSS  
YSEEYLRNLLCRKKLGSSYHTSKHGFLSWARVCLRHCIYTPQPGFHLPLKLVL SATLTG  
TAIQVALLLLLVGVVPTIQKVRAGVTTDVSYLLAGFGIVLSEDKQEVVELVKHHLWALE  
VCYISALVLSCLLTFVLVLMRSLVTHRTNLRALHRGAALDLSPLHRSPHPSRQAIFCWMS  
10 FSAYQTAFICLGLLVQQIIFFLGTTALAFVLMPVLHGRNLLLFRSLESSWPFWLTAL  
AVILQNMAAHWVFLETHDGHPQLTNRRVLYAATFLLFPLNVLVGAIVATWRVLLSALYN  
AIHLGQMDLSLLPPRAATLDPGYTYRNFLKIEVSQSHPAMTAFCSLLLQAQSLLPRTM  
AAPQDSL RPGEDEGMQLLQTKDSMAKGARPGASRGRARWGLAYTLLHNPTLQVFRKTA  
LLGANGAQP

15 Important features of the protein:

Signal peptide:

none

20 Transmembrane domain:

54-71

93-111

140-157

197-214

291-312

356-371

425-444

464-481

505-522

Motif name: N-glycosylation site.

8-12

35 Motif name: N-myristoylation site.

50-56

167-173

232-238

308-314

332-338

516-522

618-624

622-628

631-637

652-658

Motif name: Prokaryotic membrane lipoprotein lipid attachment  
50 site.

355-366

Motif name: ATP/GTP-binding site motif A (P-loop).

123-131

FIGURE 8

# Stra6 Variant Clones

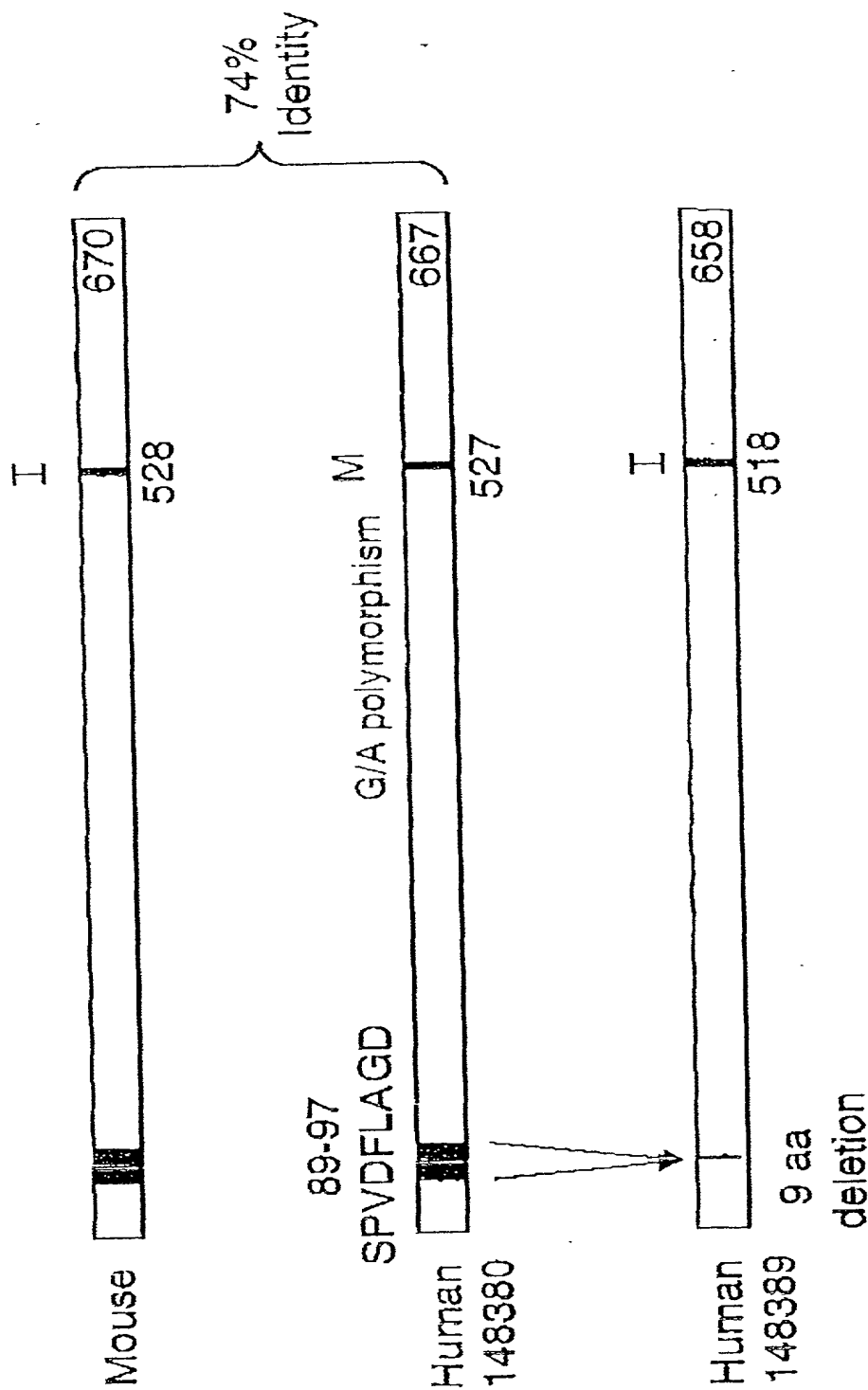
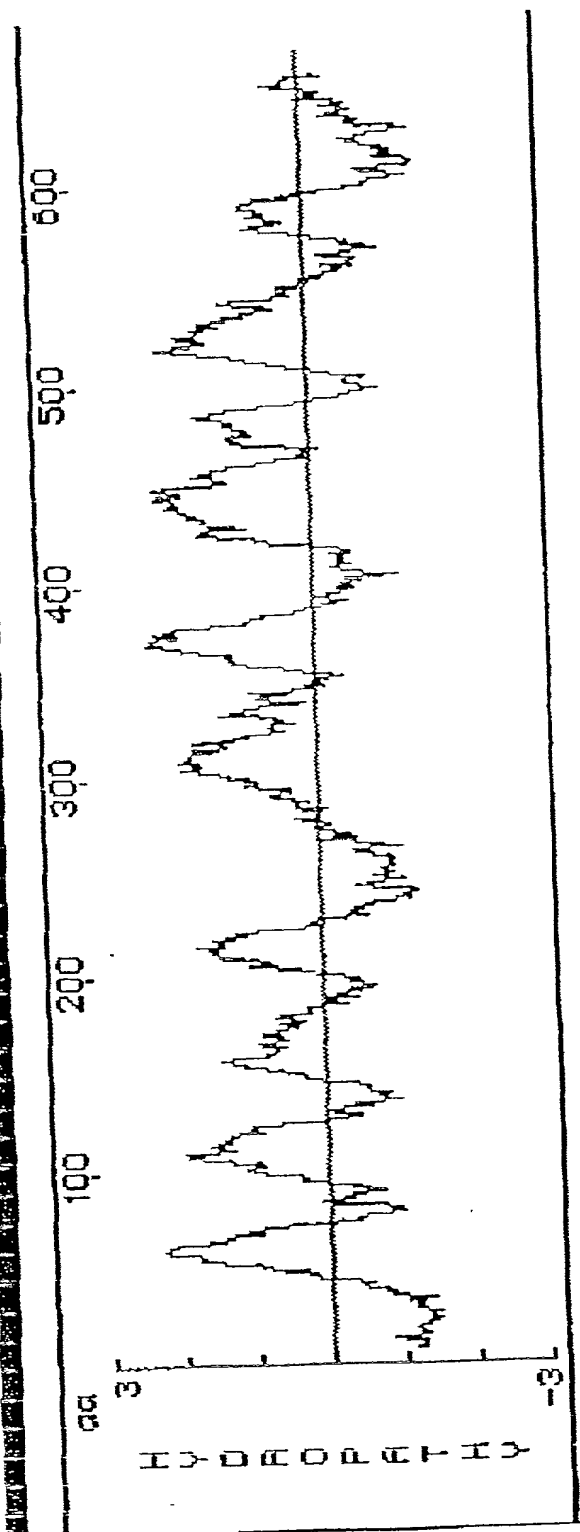


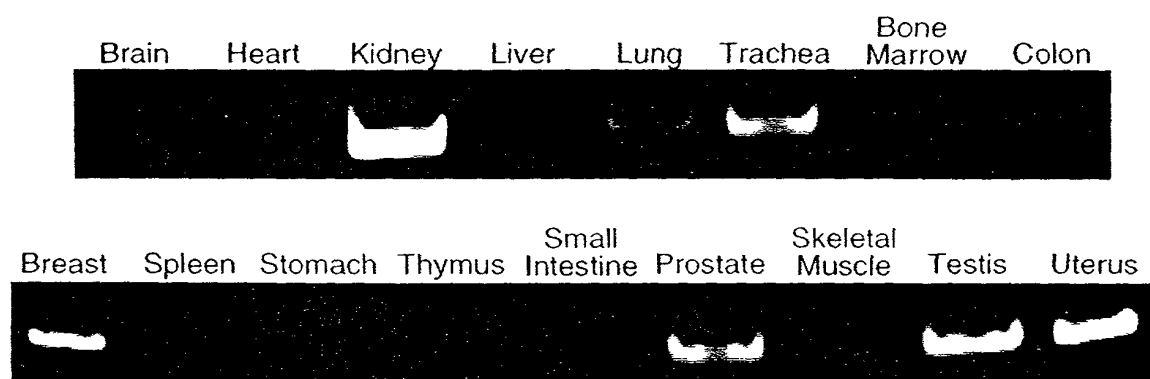
FIGURE 9

## Hydrophobicity Plot of Human Stra6



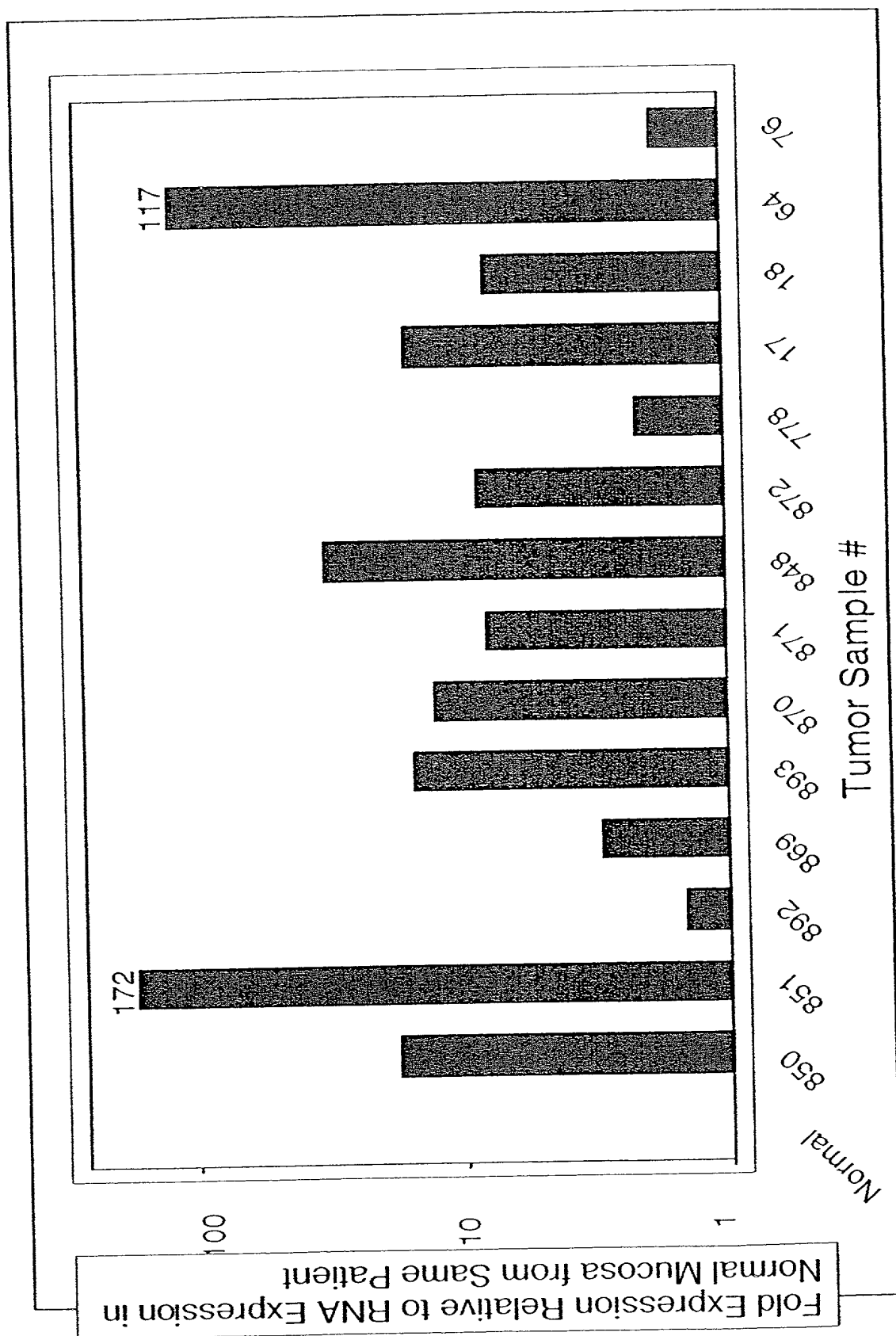
- 3 kb mRNA
- 667 Amino Acids --> 50% Residues Hydrophobic
- 73.5 kDa Protein
- 9 Potential Transmembrane Domains

FIGURE 10



# Stra6 RNA Expression in Human Colon Tumor Tissue

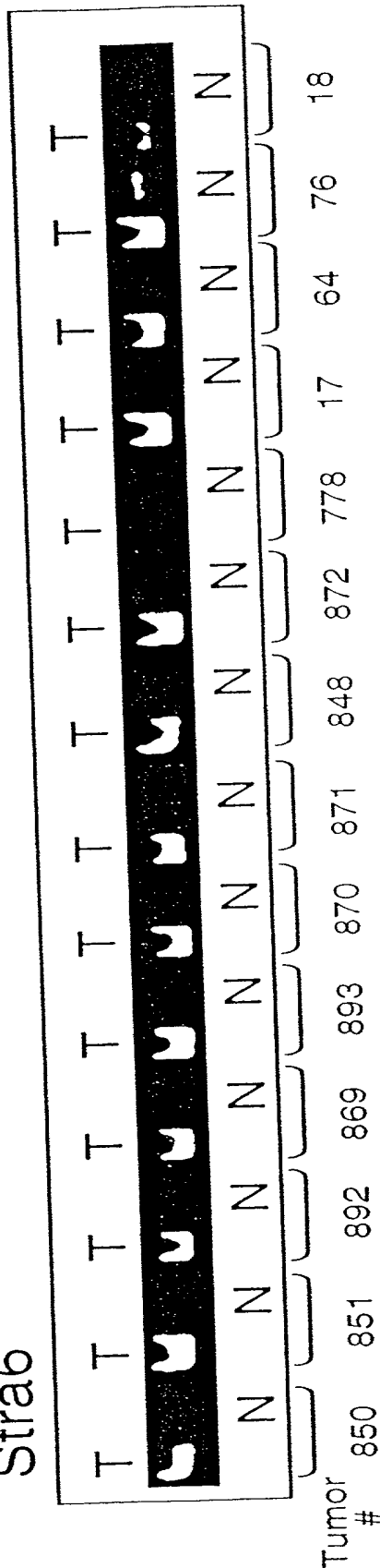
FIGURE 11



# Stra6 RNA Expression in Human Colon Tumor Tissue vs Normal Mucosa From the Same Patient

Taqman Product Analysis After 40 Cycles

Stra6



GAPDH



FIGURE 12B

C

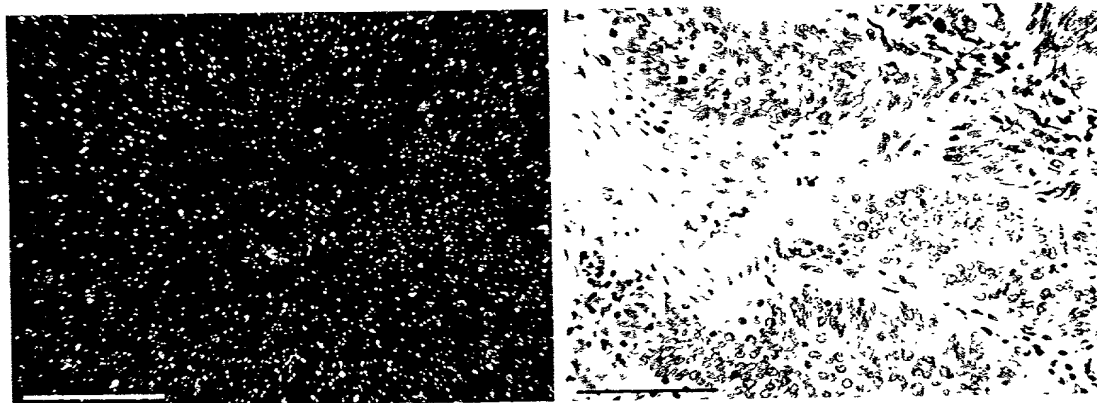
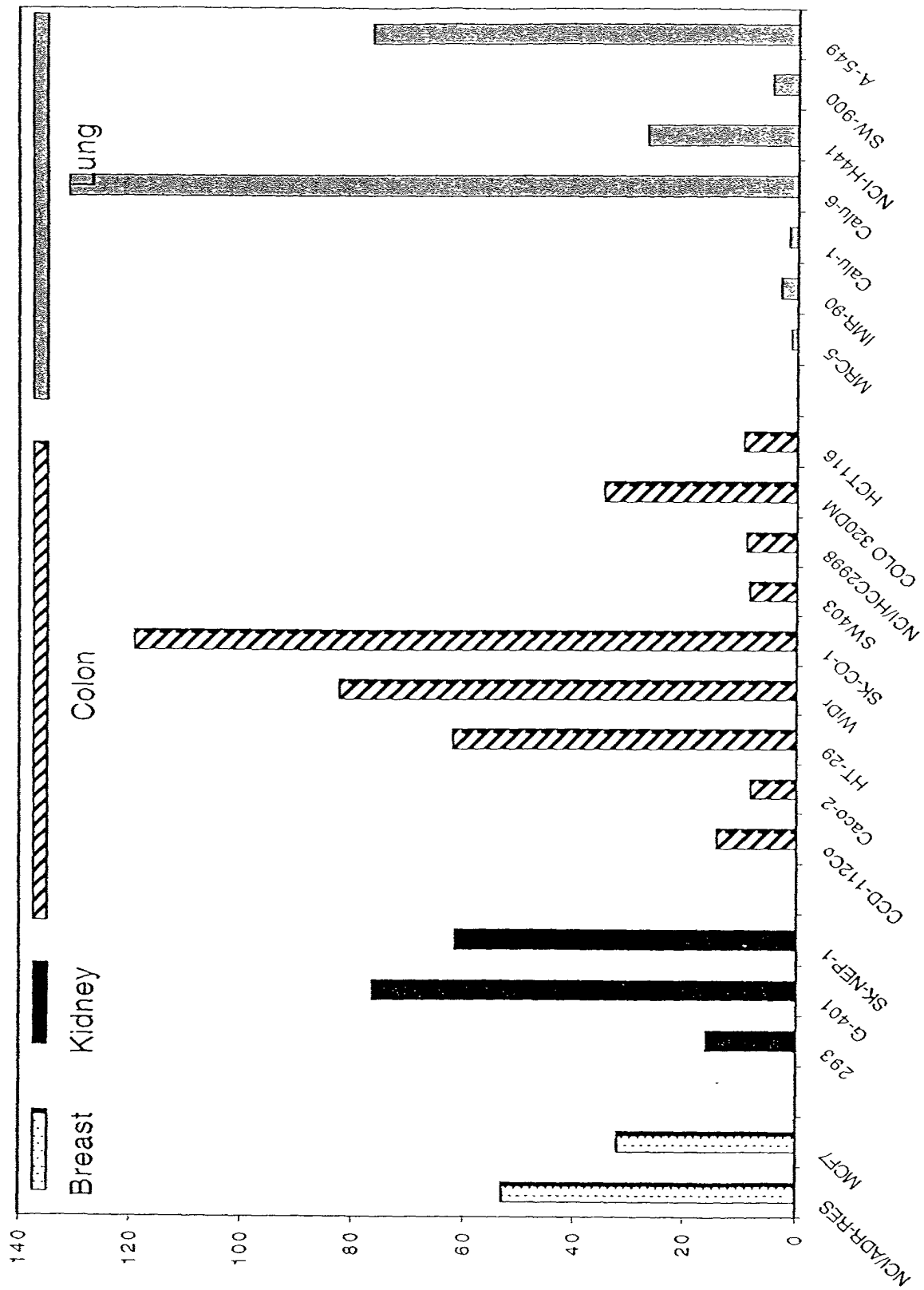
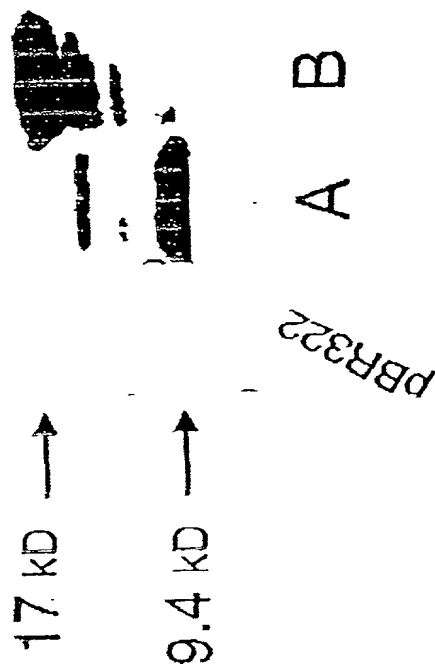
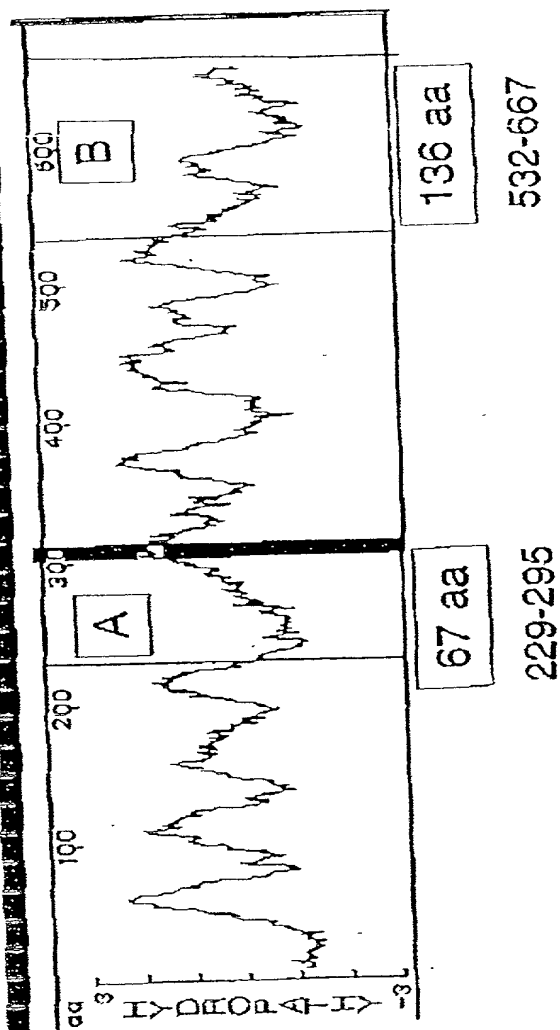




FIGURE 13



# Stra6 Peptide Expression in E. coli Poly-His Cleavable Leader at N-Terminus



500 ml Culture  
15 µl/Lane

Estimate:  
~100 µg/ml

~50 mg/500 ml

Dan Yansura

FIGURE 14

FIGURE 15

# Stra6 RNA Expression in Human Colon Carcinoma Cells +/- Retinoic Acid

TM#75 (2/28/00)

VD3 - vitamin D3 (1µM); ATRA - all-trans-retinoic acid (1 µM)

9cRA - 9-cis-retinoic acid (1 µM)

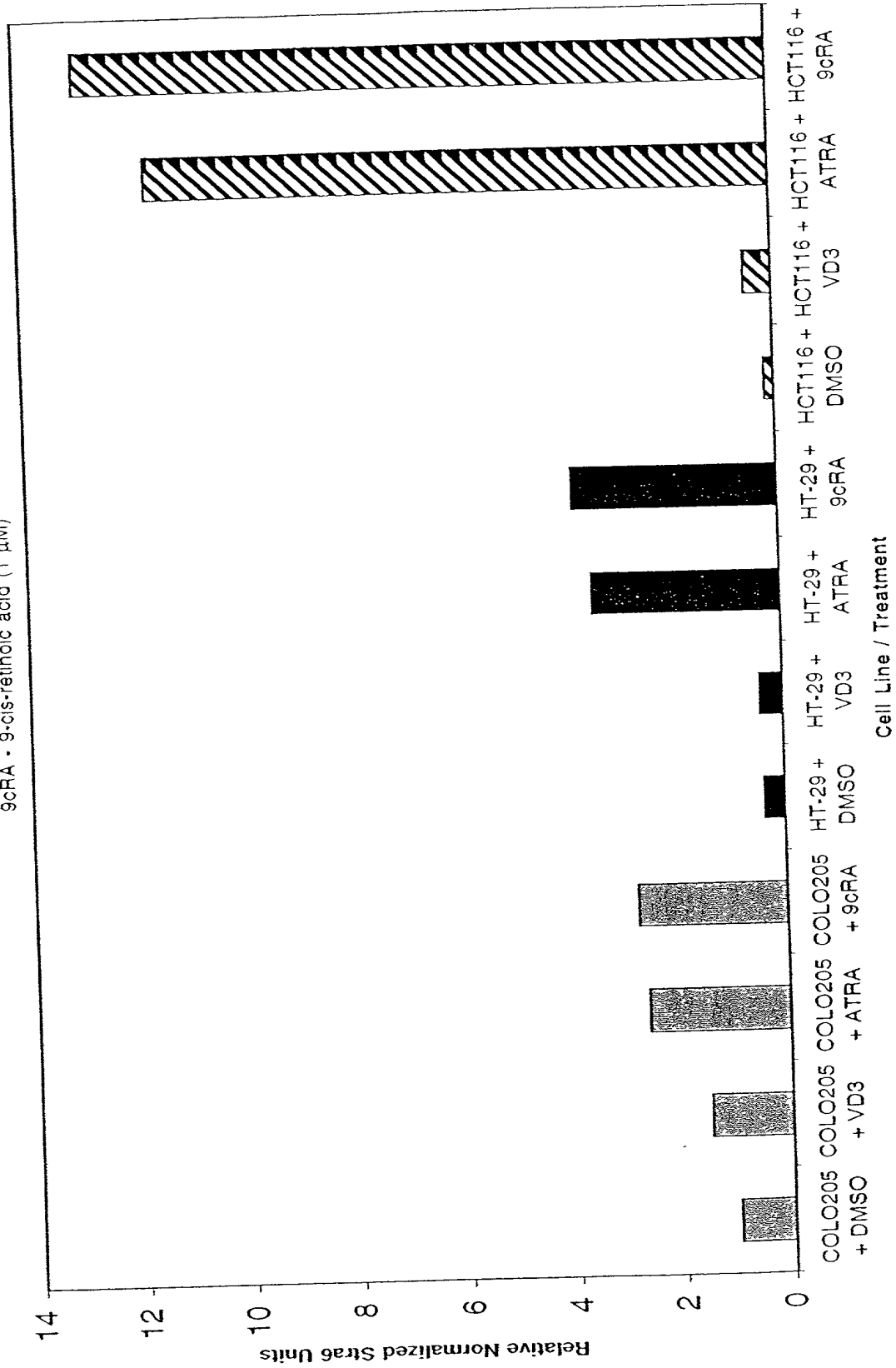


FIGURE 16

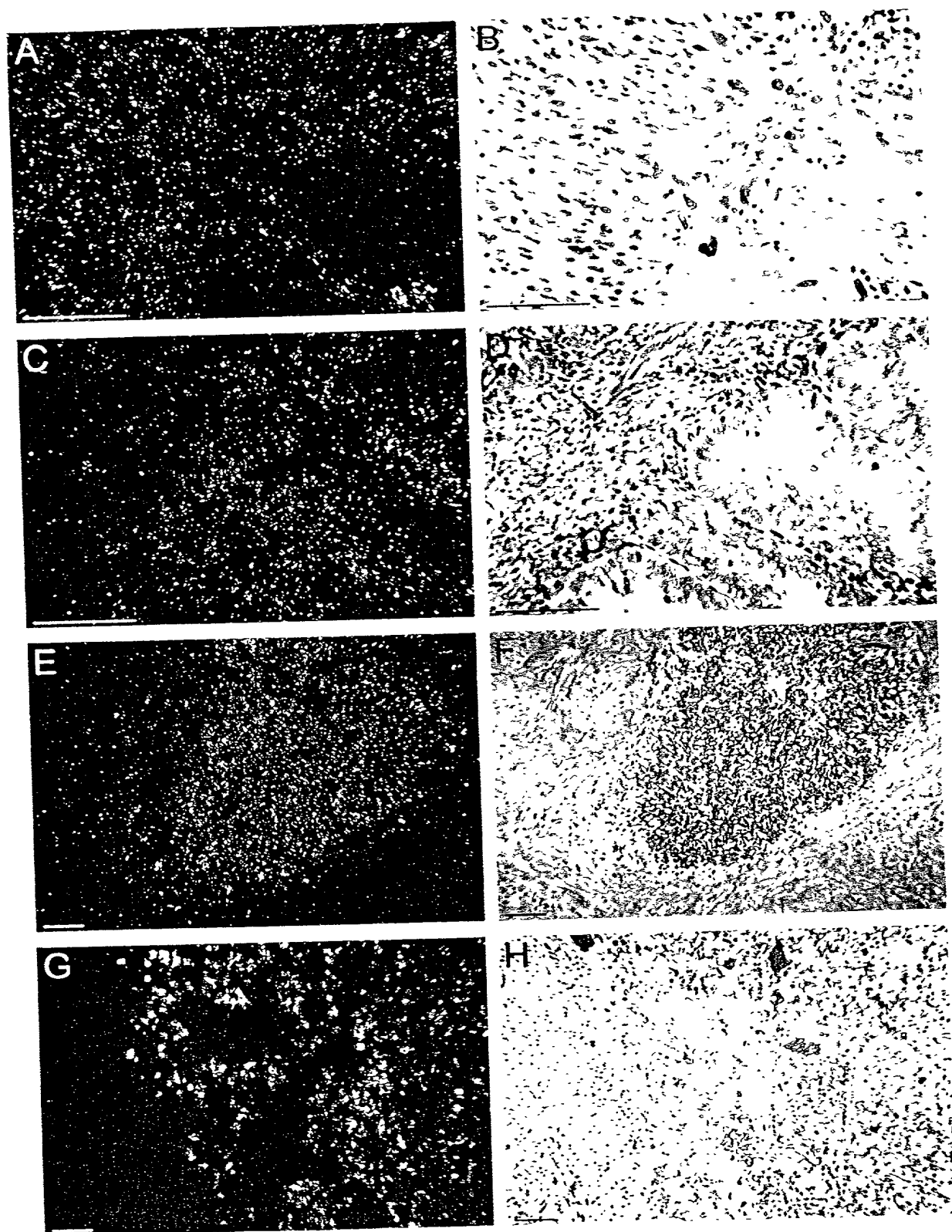


FIGURE 17

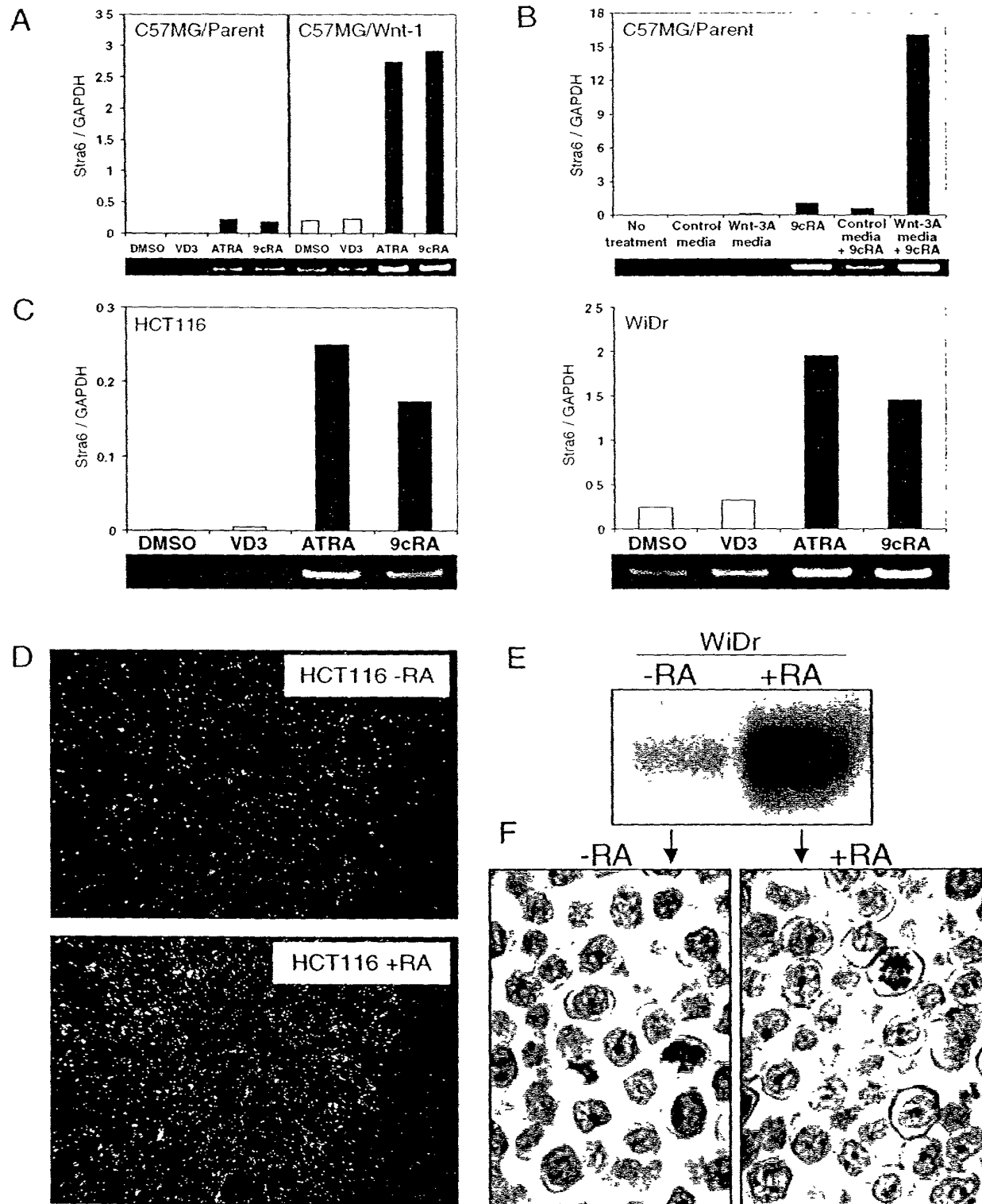


FIGURE 18

